Event-Driven Optimal Control of Continuous Petri Nets

Jorge Júlvez * † Alberto Bemporad ‡ Laura Recalde † Manuel Silva †
† Dep. Informática e Ingeniería de Sistemas, Centro Politécnico Superior de Ingenieros,
Universidad de Zaragoza, María de Luna 3, E-50015 Zaragoza, Spain,
{julvez, lrecalde, silva}@unizar.es
‡ Dip. Ingegneria dell'Informazione, Università di Siena, Via Roma 56, 53100 Siena, Italy,
bemporad@unisi.it

***Abstract*— Optimally controlling a hybrid system is a challenging problem for which mainly continuous-time and discrete-time methods have been suggested. In this paper, the problem of optimal control is addressed in the framework of continuous Petri nets, a kind of hybrid systems whose state evolution is piecewise linear. The proposed approach consists of transforming the continuous Petri net into an equivalent hybrid system whose evolution is described by means of discrete-event steps. In particular, each step coincides with the occurrence of an event in the continuous Petri net. Thus, the number of steps required to know the behavior of the Petri net is minimum, while the accuracy is completely preserved. It is shown how to design a Mixed Integer Linear Programming problem in order to compute the optimal control solution of different performance criteria.**

I. Introduction

The study of hybrid systems is becoming a field of increasing importance due to its large number of applications. A hybrid system can be seen as an interaction of a continuous time system and a discrete event system. The issue of optimal control for hybrid systems has recently attracted the attention of many researches. The different approaches taken to face the problem of optimal control can be roughly divided into two groups: those using continuous-time hybrid models and those using discrete-time hybrid models. Regarding continuous-time hybrid models, the main considered issues are the study of necessary trajectories to be optimal and the computation of optimal control laws by means of Hamilton-Jacobi-Bellman equations or the maximum principle. With respect to discrete-time hybrid models, a solution to optimal control problems was proposed in [4]. Time discretization has two important drawbacks: 1) The length of the sampling period is not easy to define. There exists a tradeoff between accuracy (short sampling period) and computational speed (long sampling period). In fact, the complexity typically grows exponentially with the number of switching variables, and these, for a given time interval, are inversely proportional to the length of the sampling period. 2) It is assumed that events can occur only at time instants that are multiple of the sampling period.

Ideally, one would like to deal with a model that requires a minimum number of steps (samples) without loss of accuracy. Consider for instance the following hybrid system expressed in continuous-time:

$$\begin{cases} \dot{x} = 2 + 2 \cdot u \text{ with } u \in [-1, 1] \text{ if } x < 8 \\ \dot{x} = 1 + 1 \cdot u \text{ with } u \in [-1, 2] \text{ if } x \geq 8 \end{cases} \quad (1)$$

and discretize it with a sampling period of one time unit:

$$\begin{cases} x(k+1) = x(k) + 2 + 2 \cdot u \text{ with } u \in [-1, 1] \text{ if } x < 8 \\ x(k+1) = x(k) + 1 + 1 \cdot u \text{ with } u \in [-1, 2] \text{ if } x \geq 8 \end{cases} \quad (2)$$

Let us assume that we are interested in the time optimal control problem of driving the system from the origin, $x(0) = 0$, to the target state $x = 14$ in minimum time. An optimal control for this problem is: $u(0) = 1$ ($x(1) = 4$), $u(1) = 1$ ($x(2) = 8$), $u(2) = 2$ ($x(3) = 11$), $u(3) = 2$ ($x(4) = 14$). Now suppose that the system has not been discretized with respect to time, but that the end of each sampling period of the model coincides with the occurrence of an event, i.e., a switch in the dynamics of the system. The time elapsed between events is now a real variable q. Such a model may be classified as *event-driven*. If that model could be created the time optimal control to reach $x = 14$ would be: $u(0) = 1$ during $q(0) = 2$ time units ($x(1) = 8$), $u(1) = 2$ during $q(1) = 2$ time units ($x(2) = 14$). That is, only two steps are necessary. In fact, given the linearity of the system, its whole evolution can be derived from the state at the event instants, and therefore steps 1 and 3 of the discrete-time model could be avoided.

The idea of optimal control via event-driven models is a relatively novel approach for controlling hybrid systems (see for example [10]). In this paper, such an approach is presented and applied to the model of continuous Petri nets (PN). Continuous PNs offer a great modelling power and represent the continuous relaxation of the original discrete PNs. This relaxation allows to face the state explosion problem inherent to large discrete systems.

The rest of the paper is organized as follows: In Section II, continuous Petri nets are presented. It will be seen that under a given firing semantics, continuous Petri nets can be considered as piecewise linear systems. Moreover, it will be shown how input actions can be introduced into the model in order to control it. Section III is concerned with the transformation of a continuous Petri net into a novel event-based Mixed Logical Dynamical System (eMLD) [4], a class of hybrid models that is very suitable for optimization

* Supported by a grant from D.G.A. ref B106/2001
† Partially supported by project CICYT and FEDER TIC2001-1819 and DPI2003-06376

purposes. The most important feature of this transformation is that the obtained eMLD system switches its continuous-time dynamics if and only if an event occurs in the original continuous Petri net. In Section IV, it is shown how optimal control problems can be solved by using Mixed Integer Linear Programming techniques. Conclusions are drawn in Section V.

II. Continuous Petri nets

In the following it is assumed that the reader is familiar with Petri nets (PNs) (see [8], [5] for example). The usual PN system will be denoted as $\langle \mathcal{N}, \mathbf{m_0} \rangle$, where $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ represents the net graph and $\mathbf{m_0}$ is the initial marking. The sets P and T contain respectively the n places and s transitions of the net. Matrices Pre and $Post$ contain respectively the input and output arc weights of transitions. All the Petri net systems to be considered are *continuous*. A continuous systems is understood as relaxation of a *discrete* system. The main difference between continuous and discrete PNs is in the firing count vector and consequently in the marking, which in discrete PNs are restricted to be in the naturals, while in continuous PNs are relaxed into the non-negative real numbers. The marking of a place can be seen as an amount of fluid being stored, and the firing of a transition can be considered as a flow of this fluid going from a set of places (input places) to another set of places (output places).

Considering an untimed PN system, it will be said that transition t is *enabled* at marking $\mathbf{m}$ iff for every $p \in {}^\bullet t$, $\mathbf{m}[p] > 0$, and its *enabling degree* is $\mathrm{enab}(t, \mathbf{m}) = \min_{p \in {}^\bullet t} \{\mathbf{m}[p] / \mathbf{Pre}[p, t]\}$. The firing of t in a certain amount $\alpha \leq \mathrm{enab}(t, \mathbf{m})$ leads to a new marking $\mathbf{m}' = \mathbf{m} + \alpha \cdot \mathbf{C}[P, t]$, where $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ is the token flow or incidence matrix. Hence, as in discrete systems, the state (or fundamental) equation ($\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}$) summarizes the way the marking evolves.

A. Timed systems

For the timing interpretation of *continuous* PNs a first order (or deterministic) approximation of the discrete case [9] will be used, assuming that the delays associated to the firing of transitions can be approximated by their mean values. Then, the state equation has an explicit dependence on time $\mathbf{m}(\tau) = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}(\tau)$. Deriving with respect to time, $\dot{\mathbf{m}}(\tau) = \mathbf{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$ is obtained. Let us denote $\mathbf{f} = \dot{\boldsymbol{\sigma}}$, since it represents the flow of the transitions.

Different semantics have been defined for continuous PNs, the most important being *infinite servers* [9] and *finite servers* [1]. In this paper finite servers semantics will be considered. Under finite server semantics, the flow vector, $\mathbf{f}$, is piecewise constant, and therefore the marking evolution is piecewise linear. The vector $\mathbf{f}$ keeps constant until an event occurs. Between events, the system is said to be at a *invariant behavior state* (IB - state) [1]. In continuous PNs an event occurs only when a place becomes empty. Thus, the number of potential IB - states equals the number of sets of places that can be empty. In principle, each place can be empty or not empty, hence the number of potential IB - states for a general system with n places is 2^n. However, the number of potential IB - states is usually not so big, since initially marked p-semiflows ([8], [5]) cannot be emptied.

Under finite firing semantics, every transition, t, has associated a real parameter $\boldsymbol{\lambda}[t] > 0$ that is the maximum flow of the transition. Intuitively, if a transition is seen as a valve through which a fluid passes, $\boldsymbol{\lambda}$ can be seen as the maximum flow admitted by the valve. In contrast to [3] no lower bound for the flow of the transitions is specified, thus the minimum flow of every transition is 0. A transition, t, is *strongly enabled* if every input place of t is marked or t has no input places. If t is strongly enabled then $\mathbf{f}[t] = \boldsymbol{\lambda}[t]$. A transition, t, is *weakly enabled* if one or more input places of t are empty and receiving an input flow from some of their input transitions, and the rest of input places of t are marked. The flow of the weakly enabled transitions has to be defined in such a way that the nonnegativity of the marking is assured. The computation of an admissible $\mathbf{f}$ is not trivial when several empty places appear. In [2], an iterative algorithm is suggested to compute one admissible $\mathbf{f}$. In this paper, $\mathbf{f}$ will be computed in a similar way to [3] where the set of admissible $\mathbf{f}$ is characterized by a set of linear inequalities. We will choose an $\mathbf{f}$ that fulfils the system of linear inequalities and maximizes $\sum_{i=1}^{s} \mathbf{f}[t_i]$. We will consider that all the transitions have the same priority. If a transition is neither strongly nor weakly enabled its flow is 0.

Let us consider the system in Figure 1(a). The only input place of t_1 is marked, hence it is strongly enabled and $\mathbf{f}[t_1] = \boldsymbol{\lambda}[t_1] = 2$. The evolution of $\mathbf{m}[p_1]$ is given by $\dot{\mathbf{m}} = \boldsymbol{\lambda}[t_2] - \boldsymbol{\lambda}[t_1] = -1$. At time 1, p_1 becomes empty, i.e., an event occurs, and t_1 becomes weakly enabled. Now, the maximum flow admitted by t_1 is 1, a greater flow will cause $\mathbf{m}[p_1]$ to be negative. Being $\mathbf{f}[t_1] = 1$, p_1 remains empty. Now p_1 can be seen as a tube instead of a deposit and no more events occur. For arbitrary values of $\boldsymbol{\lambda}[t_1]$ and $\boldsymbol{\lambda}[t_2]$, the flow of t_1 when p_1 is empty is defined as $\mathbf{f}[t_1] = \min(\boldsymbol{\lambda}[t_1], \boldsymbol{\lambda}[t_2])$.

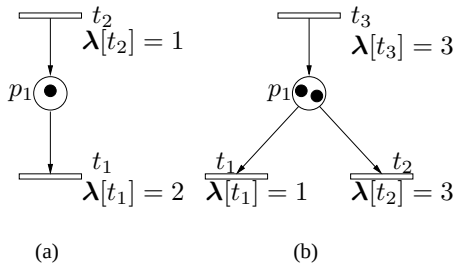


Fig. 1. (a) Transition t_1 becomes weakly enabled at $\tau = 1$. (b) Transitions t_1 and t_2 become weakly enabled at $\tau = 2$.

Transitions t_1 and t_2 of the system in Figure 1(b) are strongly enabled for the given initial marking $\mathbf{m}[p_1] = 2$. After two time units, p_1 becomes empty and t_1, t_2 become

weakly enabled. At this point, it has to be decided how to split the input flow, $\boldsymbol{\lambda}[t_3]$, coming into p_1. Since we are considering that t_1 and t_2 have the same priority, half of the flow should be routed to t_1 and half to t_2. Unfortunately, the maximum flow of t_1, $\boldsymbol{\lambda}[t_1] = 1$, is smaller than $\boldsymbol{\lambda}[t_3]/2$. To solve this situation, the flow that cannot be consumed by t_1, $\boldsymbol{\lambda}[t_3]/2 - \boldsymbol{\lambda}[t_1]$, is routed to t_2. This results in $\mathbf{f}[t_1] = 1$ and $\mathbf{f}[t_2] = 2$. The explicit analytic expressions for $\mathbf{f}[t_1]$ and $\mathbf{f}[t_2]$ with arbitrary $\boldsymbol{\lambda}[t_1]$, $\boldsymbol{\lambda}[t_2]$ and $\boldsymbol{\lambda}[t_3]$ when $\mathbf{m}[p_1] = 0$ are $\mathbf{f}[t_1] = \min(\boldsymbol{\lambda}[t_3]/2 + \max(0, \boldsymbol{\lambda}[t_3]/2 - \boldsymbol{\lambda}[t_2]), \boldsymbol{\lambda}[t_1])$ and $\mathbf{f}[t_2] = \min(\boldsymbol{\lambda}[t_3]/2 + \max(0, \boldsymbol{\lambda}[t_3]/2 - \boldsymbol{\lambda}[t_1]), \boldsymbol{\lambda}[t_2])$.

B. Controlled systems

Control actions can be introduced into the model in order to modify the *autonomous* timed evolution of the system. In continuous PNs these actions are applied to the transitions of the net. A transition t is *controllable* when its flow can be slowed down in a quantity that depends on the input, $\mathbf{u}[t]$, applied to it. The value $\mathbf{u}[t]$ is positive and upper limited by $\boldsymbol{\lambda}[t]$. An action $\mathbf{u}[t]$ on the transition t can be seen as if the valve associated to t was closed in an amount $\mathbf{u}[t]$. The way of computing $\mathbf{f}$ is analogous to the one shown in Subsection II-A, being now the maximum flow allowed by t, $\boldsymbol{\lambda}[t] - \mathbf{u}[t]$. Hence, if transition t is strongly enabled then $\mathbf{f}[t] = \boldsymbol{\lambda}[t] - \mathbf{u}[t]$. If t is weakly enabled $\mathbf{f}[t]$ will be computed considering $\boldsymbol{\lambda}[t] - \mathbf{u}[t]$ the upper bound for the flow of t. If t is neither strongly nor weakly enabled $\mathbf{f}[t] = 0$.

To show how input actions modify the evolution of a system, let us apply the input action $\mathbf{u}[t_1] = 0.5$ to the system in Figure 1(a). Since, transition t_1 is initially strongly enabled, its flow will be $\mathbf{f}[t_1] = \boldsymbol{\lambda}[t_1] - \mathbf{u}[t_1] = 1.5$. After two time units p_1 becomes empty. Hence, the maximum flow allowed by t_1 is the input flow coming to p_1, that is 1. Now, every input action on t_1 ranging from 0 to 1 has no effect on the system evolution. However, if $\mathbf{u}[t_1]$ is greater that 1, the flow of t_1 will be slowed down. For example, if $\mathbf{u}[t_1] = 1.5$, the flow of t_1 will be $\mathbf{f}[t_1] = 0.5$, and consequently p_1 will start to fill. The analytic expression for $\mathbf{f}[t_1]$ with arbitrary $\boldsymbol{\lambda}[t_1]$ and $\boldsymbol{\lambda}[t_2]$ when p_1 is empty is $\mathbf{f}[t_1] = \min(\boldsymbol{\lambda}[t_1] - \mathbf{u}[t_1], \boldsymbol{\lambda}[t_2] - \mathbf{u}[t_2])$.

Similarly, for the system in Figure 1(b), if $\mathbf{m}[p_1] > 0$ then $\mathbf{f}[t_1] = \boldsymbol{\lambda}[t_1] - \mathbf{u}[t_1]$ and $\mathbf{f}[t_2] = \boldsymbol{\lambda}[t_2] - \mathbf{u}[t_2]$. If $\mathbf{m}[p_1] = 0$ then $\mathbf{f}[t_1] = \min((\boldsymbol{\lambda}[t_3] - \mathbf{u}[t_3])/2 + \max(0, (\boldsymbol{\lambda}[t_3] - \mathbf{u}[t_3])/2 - (\boldsymbol{\lambda}[t_2] - \mathbf{u}[t_2])), \boldsymbol{\lambda}[t_1] - \mathbf{u}[t_1])$ and $\mathbf{f}[t_2] = \min((\boldsymbol{\lambda}[t_3] - \mathbf{u}[t_3])/2 + \max(0, (\boldsymbol{\lambda}[t_3] - \mathbf{u}[t_3])/2 - (\boldsymbol{\lambda}[t_1] - \mathbf{u}[t_1])), \boldsymbol{\lambda}[t_2] - \mathbf{u}[t_2])$.

III. Modelling continuous Petri nets as event-driven Mixed Logical Dynamical systems

A. Mixed Logical Dynamical systems

Mixed logical dynamical (MLD) systems [4] are computationally oriented representations of hybrid systems. They consist of a set of linear equalities and inequalities involving both real and Boolean $(0, 1)$ variables. An MLD system is described by the following relations:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B_1\mathbf{u}(k) + B_2\boldsymbol{\delta}(k) + B_3\boldsymbol{z}(k) + B_5 \quad (3)$$

$$\mathbf{y}(k) = C\mathbf{x}(k) + D_1\mathbf{u}(k) + D_2\boldsymbol{\delta}(k) + D_3\mathbf{z}(k) + D_5 \quad (4)$$

$$E_2\boldsymbol{\delta}(k) + E_3\mathbf{z}(k) \leq E_1\mathbf{u}(k) + E_4\mathbf{x}(k) + E_5 \quad (5)$$

where $\mathbf{x} \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$ is a vector of continuous and binary states, $\mathbf{u} \in \mathbb{R}^{m_r} \times \{0,1\}^{m_b}$ are the inputs, $\mathbf{y} \in \mathbb{R}^{p_r} \times \{0,1\}^{p_b}$ are the outputs, $\boldsymbol{\delta} \in \{0,1\}^{r_b}$, $\mathbf{z} \in \mathbb{R}^{r_r}$ represent auxiliary binary and continuous variables, respectively, and $A, B_1, B_2, B_3, C, D_1, D_2, D_3, E_1, E_2, E_3, E_4, E_5$ are matrices of suitable dimensions. Given the current state $\mathbf{x}(k)$ and input $\mathbf{u}(k)$, the evolution of (3)-(5) is determined by solving $\boldsymbol{\delta}(k)$ and $\mathbf{z}(k)$ from (5) and then updating $\mathbf{x}(k+1)$ and $\mathbf{y}(k)$ from (3) and (4). It is assumed that the system (3)-(5) is completely *well-posed* [4], which means that for all $\mathbf{x}(k)$, $\mathbf{u}(k)$ within a given bounded set the variables $\boldsymbol{\delta}(k)$, $\mathbf{z}(k)$ are defined by (5) in a unique way.

Several conversion laws exist that allow to transform logic relations into mixed-integer inequalities. For example the threshold condition

$$[\delta = 1] \leftrightarrow [a \cdot x + b \cdot u + c \geq 0] \quad (6)$$

where $\delta \in \{0,1\}$, $a, b, c, x, u \in \mathbb{R}$ can be equivalently expressed as:

$$\begin{cases} a \cdot x + b \cdot u + c < M \cdot \delta \\ a \cdot x + b \cdot u + c \geq m \cdot (1 - \delta) \end{cases} \quad (7)$$

where M, m are upper and lower bounds, respectively, on $a \cdot x + b \cdot u + c$. In a similar way, the semantics of the common relation

$$\text{IF } \delta \text{ THEN } z = a_1 \cdot x + b_1 \cdot u + c_1 \text{ ELSE } z = a_2 \cdot x + b_2 \cdot u + c_2 \quad (8)$$

which links Boolean to continuous variables, can be expressed with the following set of inequalities:

$$\begin{cases} (m_2 - M_1) \cdot \delta + z \leq a_2 \cdot x + b_2 \cdot u + c_2 \\ (m_1 - M_2) \cdot \delta - z \leq -a_2 \cdot x - b_2 \cdot u - c_2 \\ (m_1 - M_2) \cdot (1 - \delta) + z \leq a_1 \cdot x + b_1 \cdot u + c_1 \\ (m_2 - M_1) \cdot (1 - \delta) - z \leq -a_1 \cdot x - b_1 \cdot u - c_1 \end{cases} \quad (9)$$

where $\delta \in \{0,1\}$, $a_1, b_1, c_1, a_2, b_2, c_2, x, u, z \in \mathbb{R}$ and M_i, m_i are upper and lower bounds on $a_i \cdot x + b_i \cdot u + c_i$, $i = 1, 2$. Further details on these and other conversions can be seen for example in [7]. These transformations allow MLD systems to model a great variety of hybrid systems.

Usually, in an MLD system k represents a time-step counter, and the length of the time step is constant. It is common that the shorter the time step the greater the accuracy of the model. Clearly, the main drawback of having a very short time step is that many steps may be required to study the evolution of the system during a given time interval. Assuring good accuracy while minimizing the number of steps is a good criterion to choose the length of the time step.

B. Continuous Petri net as event-driven Mixed Logical Dynamical systems

We will show how to use the MLD transformation machinery (see e.g. [11]) in order to describe the behavior of a *non-controlled* continuous PN. In a non-controlled system no input actions are considered, and therefore, the value of $\mathbf{f}$ is constant between events and depends only on the IB - state of the net. By treating each step k as the occurrence of an event in the continuous PNrather than the elapse of a sampling period, we will transform the continuous PN into an event-based mixed logical dynamical (eMLD) system. In other words, after each step a place becomes empty. Observe that this approach has two interesting advantages:

- Event-discretization does not imply loss of accuracy: The marking evolution of a continuous PN is linear between events, and so it can be determined from the marking of the net at the event instants.
- The number of steps is minimized: A step happens only when it is really required (an event happens).

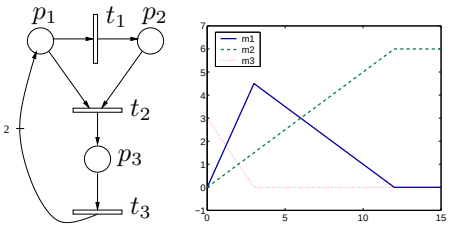
Clearly, this implies that the length of the period cannot be constant but depends on the event instants, therefore making the eMLD system we are dealing with a truly *event-driven* system, rather than a time-drive one. The time period will be a real variable of the eMLD system expressing the length of the interval between two events.

The steps to convert a continuous PN into the aforementioned eMLD system are the following:

1) *Identify the potential IB - states of the continuous PN.* That is, the potential dynamics that may rule the evolution of the system. For example, the system in Figure 4 has four potential IB - states: a) $\mathbf{m}[p_1] > 0$, $\mathbf{m}[p_2] > 0$, b) $\mathbf{m}[p_1] > 0$, $\mathbf{m}[p_2] = 0$ c) $\mathbf{m}[p_1] = 0$, $\mathbf{m}[p_2] > 0$ d) $\mathbf{m}[p_1] = 0$, $\mathbf{m}[p_2] = 0$.
2) *Describe the behavior of the PN under each IB - state.* This is equivalent to define the flow of the transitions under each IB - state. See subsection II-A for the expressions defining the flows of the transitions of the system in Figure 1(b).
3) *Define the evolution of the marking.* As seen in Subsection II-A, the derivative of the marking is given by the incidence matrix multiplied by the flows of the transitions. This has to be included in the set of equations of the eMLD system. For the system in Figure 1(b), the equation defining the evolution of the marking is: $\mathbf{m}(k+1) = \mathbf{m}(k)+q\cdot(\mathbf{f}[t_3]-\mathbf{f}[t_1]-\mathbf{f}[t_2])$, where q is the real variable storing the time elapsed between events k and $k+1$.
4) *Force that at least one place becomes empty at the occurrence of the next event.* To achieve this, a Boolean variable per place can be defined. The Boolean variable becomes true iff the place at event k is marked and becomes empty after q time units. By means of equation (5) it is easy to force that the sum of these Boolean variables is positive, i.e., at least one place becomes empty.

The task of transforming and event-driven model describing a continuous PN into an eMLD system in the form of (3) is greatly eased by HYSDEL [11] (HYbrid System DEscription Language). Basically, HYSDEL allows one to describe a hybrid system in textual form and outputs the matrices of the equivalent MLD form.

Consider the system in Figure 2(a) with $\boldsymbol{\lambda} = (1.5\ 1\ 2)$ and $\mathbf{m_0} = (0\ 0\ 3)$. The system is transformed into eMLD form following the tasks described above. Only two steps are necessary to reach the steady state marking. The length of the first two intervals is respectively $q(1) = 3$ and $q(2) = 9$. The evolution of the system is depicted in Figure 2(b). After the second step, i.e., event, p_1 and p_3 become empty and the system dies, that is, the flow of every transition is zero in the steady state.



(a) (b)

Fig. 2. (a) A continuous PN system. (b) Its marking evolution.

IV. Optimal control using Mixed Integer Linear Programming

A. Obtaining a Mixed Integer Linear Programming

The eMLD system obtained in the previous section must be slightly modified in order to take into account the effect of the control actions in the marking evolution. As explained in Subsection II-B, when a transition t is controllable its flow, $\mathbf{f}[t]$, depends on the input action $\mathbf{u}[t]$ applied to it. We will assume that the control actions are constant between events. Notice that this constraint is not very strong since the effect of a non constant $\mathbf{u}[t]$ is equivalent to the effect of a constant $\mathbf{u}[t]$ with the same integral.

Let us consider again the system in Figure 1(a) with $\mathbf{m_0}[p_1] = 1$. Suppose that transition t_1 is controllable. Now, the flow of t_1 is $\mathbf{f}[t_1] = \boldsymbol{\lambda}[t_1] - \mathbf{u}[t_1]$ and the marking evolution of p_1 from event k to $k+1$ is $\mathbf{m}(k+1) = \mathbf{m}(k)+q\cdot(\mathbf{f}[t_2]-\mathbf{f}[t_1]) = \mathbf{m}(k)+q\cdot(\boldsymbol{\lambda}[t_2]-(\boldsymbol{\lambda}[t_1]-\mathbf{u}[t_1]))$. Thus, the equation defining $\mathbf{m}(k+1)$ becomes nonlinear since both q and $\mathbf{u}[t_1]$ are real variables. Such an equation cannot be included directly in an eMLD system. A way of overcoming this situation is to define a new real variable $\mathbf{h}[t_1]$ as follows: $\mathbf{h}[t_1] = q\cdot\mathbf{u}[t_1]$. Hence, $\mathbf{f}[t_1] = q\cdot\boldsymbol{\lambda}[t_1] - \mathbf{h}[t_1]$. That is, there does not exist an explicit input variable but two real variables q and $\mathbf{h}[t_1]$ from which the value of $\mathbf{u}[t_1]$ can be obtained.

The input action on a controllable transition, $\mathbf{u}[t]$, must be bounded by 0 and $\boldsymbol{\lambda}[t]$. Equation (5) can be used to express

these bounds. For example, for the system in Figure 1(a) the equation $0 \leq \mathbf{h}[t_1] \leq q \cdot \boldsymbol{\lambda}[t_1]$ must be added.

The following step after introducing the input actions into the eMLD system is to define the function to be optimized. We will focus on linear optimization functions. The optimization variables that can be included in that function are: q, $\mathbf{h}$, $\mathbf{m}$ or any other Boolean or real variable that can be linearly originated from these ones. Furthermore, if desired, the eMLD formalism allows one to add constraints on the mentioned variables by means of Equation (5). This all leads to a multivariable optimal control problem that can be expressed as:

$$\min \mathbf{f} \cdot [\mathbf{m}(0), ., \mathbf{m}(N), \mathbf{h}(0), ., \mathbf{h}(N), \mathbf{z}(0), ., \mathbf{z}(N), \boldsymbol{\delta}(0), ., \boldsymbol{\delta}(N)]$$
$$s.t. \begin{cases} \mathbf{m}(k+1) = A\mathbf{m}(k) + B_1\mathbf{h}(k) + B_2\boldsymbol{\delta}(k) + B_3\mathbf{z}(k) + B_5 \\ E_2\boldsymbol{\delta}(k) + E_3\mathbf{z}(k) \leq E_1\mathbf{u}(k) + E_4\mathbf{x}(k) + E_5 \end{cases} \tag{10}$$

Note that the duration of each period is stored in the real variable q, that is part of the vector of real auxiliary variables $\mathbf{z}$. This control problem is defined for a horizon of N steps. It can be seen as a Mixed Integer Linear Programming (MILP) problem where the integer variables can only be 0 or 1. See [6] for a review of methods to solve MILP problems. The solution of the programming problem contains the input actions that have to be applied to the continuous PN in order to optimally control it.

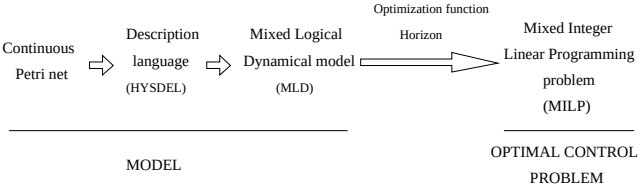


Fig. 3. Obtaining an MILP problem from a continuous PN.

Figure 3 shows the steps that have been followed to obtain an MILP problem from the initial continuous PN. In contrast to the eMLDs described in Subsection III-B the eMLDs in this Section do not represent the free evolution of the system but its controlled evolution. The controlled system will evolve in such a way that the optimization function is minimized/maximized. Events will happen without being forced so that the system behaves optimally. Therefore it is not necessary to force the occurrence of events when writing the eMLDs, i.e., step 4 in Subsection III-B can be skipped.

B. Optimality Criteria

The optimization examples of this Subsection show how different kinds of optimal control problems are solved by means of the explained event-driven approach. The control problems have to do with reaching a target marking in minimum time, i.e., time optimal control, maximizing the steady state throughput and maximizing an optimization function in which several different parameters are involved.

1) Time optimal control: Consider the system in Figure 4 where $\mathbf{m_0} = (4\ 2)$, $\boldsymbol{\lambda} = (2\ 4\ 3\ 2)$ and the controllable transitions are t_1 and t_2. Let us consider the optimal control problem of reaching the target marking $\mathbf{m} = (0\ 5)$ in minimum time. Hence the function to minimize in (10) is $\sum_{i=0}^{N} q(i)$. It is obtained $\mathbf{u}[t_1](0) = 0$, $\mathbf{u}[t_2](0) = 2.4286$ and the duration of the step is $q(0) = 7$. The target marking is reached in one step. If it is desired to know the steady state control at the target marking, it is only necessary to force the marking in the last period to be constant. The steady state control obtained is $\mathbf{u}[t_1] = 1$, $\mathbf{u}[t_2] = 2$.

The inclusion of auxiliary Boolean variables in (10) allows one to define more elaborated optimization functions that may be useful in real situations. For example, for the system in Figure 4, a Boolean variable per place can be defined in the following way: The Boolean variable associated to p_1 (respectively p_2) is true iff the target marking of p_1 (respectively p_2) has been reached. By using these Boolean variables, it is possible to assign different priorities to different places, e.g., to favor a certain place. Assume that a penalty of 3 (respectively 1) units is obtained per time unit elapsed without reaching $\mathbf{m}[p_1] = 0$ (respectively $\mathbf{m}[p_2] = 5$). An optimal control problem that minimizes the sum of penalties yields the following control: $\mathbf{u}[t_1](0) = 0$, $\mathbf{u}[t_2](0) = 1$ during the first step of $q(0) = 2$ units reaching $\mathbf{m}(1) = (0\ 0)$ and $\mathbf{u}[t_1](1) = 0$, $\mathbf{u}[t_2](1) = 3$ during the second step of $q(1) = 5$ units reaching $\mathbf{m}(2) = (0\ 5)$.

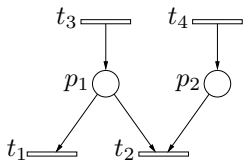


Fig. 4. A continuous PN with controllable transitions t_1 and t_2.

Consider the system in Figure 2(a) with $\boldsymbol{\lambda} = (1.5\ 1\ 2)$, $\mathbf{m_0} = (6\ 0\ 0)$ with t_1 and t_3 as controllable transitions. Let us give a penalty of 1 unit per time unit per place whose target marking has not been reached. The optimal control to reach $\mathbf{m} = (0\ 4\ 1)$ that minimizes the penalty is: $\mathbf{u}[t_1](0) = 0$, $\mathbf{u}[t_3](0) = 2$ during the first step of $q(0) = 1$ units reaching $\mathbf{m}(1) = (3.5\ 0.5\ 1)$ and $\mathbf{u}[t_1](1) = 0$, $\mathbf{u}[t_3](1) = 1$ during the second step of $q(1) = 7$ units reaching $\mathbf{m}(2) = (0\ 4\ 1)$.

2) Maximization of the steady state throughput: If no control is applied to the system in Figure 2(a) with $\boldsymbol{\lambda} = (1.5\ 1\ 2)$, it reaches a steady state with null throughput, i.e., flow, for any initial marking. An interesting control problem for such non live systems and for many manufacturing systems is to maximize the throughput in the steady state. Consider that system with $\mathbf{m_0} = (6\ 0\ 0)$ and t_3 as the only controllable transition. Let us define an optimal control problem that maximizes the throughput of t_1 in the last period. The duration of the last period is required to be 10 time units. During the last period the marking must be kept constant, i.e., it is a steady state marking. Notice that the

system has a unique T-semiflow (repetitive sequence) and therefore maximizing the throughput of t_1 in the steady state is equivalent to maximizing the throughput of any of the three transitions. The obtained control is: $\mathbf{u}[t_3](0) = 2$ during $q(0) = 2.4$ reaching $\mathbf{m}(1) = (0\ 1.2\ 2.4)$ and $\mathbf{u}[t_3](1) = 1$ for the steady state ($q(1) = 10$). At that marking the flow of the transitions is 1.

3) Optimization based on several parameters: The PN in Figure 5 represents a simple manufacturing system with two lines, (t_1, t_3) and (t_2, t_4), and a shared resource p_5. Let t_1 and t_4 be the controllable transitions, $\mathbf{m_0} = (4\ 3\ 1\ 1\ 2)$ and $\boldsymbol{\lambda} = (2\ 3\ 5\ 2)$. Suppose that we are interested in reaching a steady state marking in which $\mathbf{m}[p_1] + \mathbf{m}[p_4]$ is maximum. Besides, we would like a steady state in which the flow of the transitions is as high as possible. To do this, an optimization function including markings and flows can be defined. Weights can be assigned to the markings and flows to highlight the importance of each parameter in the optimization function. For example, an optimization function that gives the same weight to the marking and the flows is: $\mathbf{m}[p_1](N) + \mathbf{m}[p_4](N) + \mathbf{f}[t_1](N) + \mathbf{f}[t_2](N)$. Due to the existing t-semiflows, in the steady state t_3 (t_4) will have the same flow than t_1 (t_2). Therefore, in the optimization function it is not necessary to include neither $\mathbf{f}[t_3]$ nor $\mathbf{f}[t_4]$.

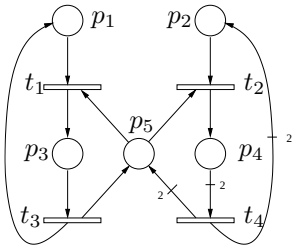


Fig. 5. A small manufacturing system.

Furthermore, it would be nice to reach the steady state marking as soon as possible. For this control problem, an adequate optimization function to be maximized is $k \cdot (\mathbf{m}[p_1](N) + \mathbf{m}[p_4](N)) + k' \cdot (\mathbf{f}[t_1](N) + \mathbf{f}[t_2](N)) - \sum_{i=0}^{N} q(i)$ with k, k' big enough to ensure that the marking of places and flows have priority (in our case it is enough $k, k' \geq 10$). The obtained control with $k = k' = 10$ is: $\mathbf{u}[t_1](0) = 2$, $\mathbf{u}[t_4](0) = 2$ with $q(0) = 0.2$ reaching $\mathbf{m}(1) = (5\ 2.4\ 0\ 1.6\ 2.4)$, $\mathbf{u}[t_1](1) = 0$, $\mathbf{u}[t_4](1) = 2$ with $q(1) = 0.8$ reaching $\mathbf{m}(2) = (5\ 0\ 0\ 4\ 0)$ that is a steady state marking with the inputs $\mathbf{u}[t_1](2) = 0$, $\mathbf{u}[t_4](2) = 0.5$.

The control problems presented in this Section were solved by a PC Pentium IV 2.6 Ghz using the GLPK (GNU Linear Programming Kit) package solver running under Matlab 6.5. The optimal solutions were always found in less than 3 minutes.

V. Conclusions

The problem of optimally controlling a class of hybrid systems has been studied. The main concern of the paper is to overcome the drawbacks that appear when considering discrete-time models. That is, loss of accuracy when using a too long period and high computational load when using a too short period.

The hybrid model on which the study has been focused is the class of continuous Petri nets under finite firing semantics. The dynamics of a continuous Petri net is piecewise linear. Events are triggered by changes in the marking (a place becomes empty) producing a switch in the system dynamics. Input actions have been explicitly introduced into this model and their effect in the evolution of the system has been defined.

The main effort is to obtain a hybrid model with an *event-based* discretization. That is, each step of the system coincides with the occurrence of an event. Given that the evolution of continuous Petri nets is linear between events, the whole trajectory of the system can be computed from the marking at event instants. In this way, the number of required steps is minimized while accuracy is completely preserved. To obtain such a hybrid model, continuous Petri nets are transformed into event-based mixed logical dynamical models with some particular features. Using the latter models and a linear optimization function is possible to find the input actions that optimally control the original continuous Petri net by solving a mixed integer linear programming problem.

References

[1] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8(1):159–188, 1998.

[2] H. Alla and R. David. A modeling and analysis tool for discrete event systems: Continuous petri net. *Performance Evaluation*, 33:175–199, 1998.

[3] F. Balduzzi, G. Menga, and A. Giua. First-order hybrid Petri nets: a model for optimization and control. *IEEE Trans. on Robotics and Automation*, 16(4):382–399, 2000.

[4] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[5] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman & Hall, 1993.

[6] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, 1995.

[7] D. Mignone. *Control and Estimation of Hybrid Systems with Mathematical Optimization*. PhD thesis, Automatic Control Laboratory-ETH, Zurich, 2002.

[8] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[9] L. Recalde and M. Silva. Petri Nets fluidification revisited: Semantics and steady state. *APII-JESA*, 35(4):435–449, 2001.

[10] B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, July 2001.

[11] F.D. Torrisi and A. Bemporad. HYSDEL — A tool for generating computational hybrid models. *IEEE Trans. Contr. Systems Technology*, 12(2), March 2004. `http://control.ethz.ch/~hybrid/hysdel`.